****

**PROJECT REPORT ON**

**Reward Based Ecommerce Application with Waste Management**

Submitted in partial fulfillment of the requirements for Software Engineering Mini Project Lab for 2nd Semester

M.Sc Computer Science

Submitted by

**Namith T**

**23MSCS17**

Under the guidance of

**Ms. Divya M O**

****

**CERTIFICATE**

This is to certify that the project titled “**Reward Based Ecommerce Application with Waste Management”** has been satisfactorily completed by **Mr. Namith T** with **Reg. No. 23MSCS17**, in partial fulfillment of the requirements for ***Software Engineering Mini Project Lab*** with course code **MCC2P2B21,** for the 2nd Semester M.Sc Computer Science course during the academic year 2023-2024 as prescribed by Bangalore North University.

**Faculty In-charge Head of the Department**

**Valued by**

Examiner 1:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date :

Examiner 2:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Centre: Kristu Jayanti College

**ACKNOWLEDGEMENT**

First of all, I would like to thank the God Almighty for all the blessings he has showered on us. Our spiritual quotient gave us more strength and motivation that helped immensely.

We would like to thank **Rev. Fr. Dr. Augustine George,** our esteemed Principal, for providing us their constant guidance and support. I would also like to thank **Rev. Fr. Lijo P Thomas,** our Vice-Principal, for providing us with the best facilities.

We are extremely thankful to our **Dr. Kumar R**, Head, Department of Computer Science (PG) for giving us the essential support in the form of allocating comfortable project hours and necessary software resources.

We would like to extend our heartfelt thanks to Ms**. Divya M O**, our project guide for providing us the necessary details related to project development and process identification enabling us to finish the project within the stipulated time.

We thank all other faculty members who helped us a lot in completing this project.

We thank our class mates, who have pointed out errors and guided us a lot and we thank each and every one who has helped us.

**SYNOPSIS**

**EcoHub - Empowering Sustainability**

EcoHub is a pioneering web-based platform dedicated to fostering sustainable living practices and environmental consciousness among individuals and communities. With a focus on promoting eco-friendly behaviors, facilitating waste management, and providing access to sustainable products and services, EcoHub aims to create a positive impact on our planet.

The platform's key features include:

1. **User Authentication and Profiles:** Seamless sign-up and login using Google accounts, coupled with personalized user profiles showcasing account details, order history, waste collection requests, and EcoCoin balance (a virtual currency for eco-transactions).
2. **Waste Collection Requests:** Empowering users to submit waste collection requests effortlessly, thereby contributing to responsible waste disposal and management.
3. **E-commerce Platform:** Offering a diverse range of eco-friendly products and services through a user-friendly e-commerce interface, promoting sustainable consumer choices.
4. **Admin Dashboard:** Equipping administrators with comprehensive tools to manage user accounts, monitor waste collection processes, analyze platform metrics, and curate product listings for a curated eco-friendly marketplace.

EcoHub's holistic approach leverages community engagement, educational resources, and collaboration to inspire and enable individuals to make environmentally conscious decisions in their daily lives. By harnessing technology for environmental stewardship, EcoHub aspires to drive meaningful change and contribute positively towards a sustainable future.

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SI. No.**  **1.**  **2.**  **3.**  **4.**  **5.**  **6.**  **7.**  **8.**  **9.**  **10.**  **11.** | **Particulars**  **Introduction**   * 1. **System Definition**   2. **Project Description**   **System Study**  **2.1 Existing System**  **2.2 Proposed System**  **2.3 Data Flow Diagram**  **2.4 ER Diagram**  **System Configuration**  **3.1 Hardware Configuration**  **3.2 Software Configuration**  **Details of Software**  **4.1 Overview of Front End**  **4.2 Overview of Back End**  **System Design**  **5.1 Architectural Design**  **5.2 Input Design**  **5.3 Output Design**  **5.4 Database Design**  **Source Code**  **Testing**  **Implementation**  **Screen Shot**  **Conclusion**  **Bibliography** | **Pg. No** |

1. **Introduction**

In an era marked by environmental concerns and the urgent need for sustainable practices, my project, "Waste Management and E-commerce Integration," endeavours to create an innovative solution that not only addresses the issue of waste management but also promotes recycling and sustainable consumer habits. This groundbreaking project combines two major components — a Waste Collection Submodule and an E-commerce Submodule — to create a comprehensive platform that encourages users to actively participate in waste reduction and recycling initiatives.

At its core, this project consists of two interconnected modules:

1. **Waste Collection Submodule:** This submodule focuses on streamlining waste management processes by allowing users to request waste collection services easily. Users can submit requests specifying the type of waste and its location, promoting responsible waste disposal practices. By simplifying the waste collection process, we aim to encourage more people to participate in recycling and waste reduction efforts.
2. **E-commerce Submodule:** The e-commerce component of this platform offers a curated selection of eco-friendly products and services. From organic foods and sustainable fashion items to reusable household goods, the e-commerce platform promotes sustainable consumer habits. By providing users with access to environmentally friendly alternatives, we aim to drive demand for sustainable products and contribute to a greener economy.

The integration of these two modules creates a synergistic platform that not only addresses waste management challenges but also educates and empowers users to make sustainable choices in their everyday lives. Through educational resources, community engagement, and seamless user experiences, my project strives to foster a culture of environmental responsibility and conscious consumption.

* 1. **System Definition**

The EcoHub project aims to create an integrated platform that facilitates waste management, e-commerce, and administrative functionalities. The system is designed to provide a seamless experience for users, administrators, and waste collection service providers. Below are the key components and functionalities of the EcoHub system:

**1. Waste Collection Module:**

* Allows users to request waste collection services by submitting requests through a simple form.
* Captures details such as the type of waste and the collection location.
* Provides administrators with a dashboard to manage and track waste collection requests.
* Enables waste collection service providers to view and process incoming requests efficiently.

**2. E-Commerce Module:**

* Offers a wide range of eco-friendly products for users to browse and purchase.
* Provides a user-friendly interface for product listing, browsing, and checkout.
* Integrates payment processing functionality to facilitate secure transactions.
* Allows users to add items to their shopping cart and review their orders before making a purchase.

**3. Administrative Module:**

* Empowers administrators with tools to manage users, products, and orders.
* Enables user management, including account creation, role assignment, and access control.
* Provides insights and analytics through dashboards to monitor system performance and user activities.
* Facilitates order management, including tracking, processing, and resolving customer inquiries.
  1. **Project Description**

EcoHub is a comprehensive web-based platform that serves as a catalyst for promoting sustainable living practices and environmental awareness. It is designed to offer a range of features and services aimed at encouraging individuals to adopt eco-friendly behaviors, manage waste responsibly, and access sustainable products and services. The platform is built around the principles of community engagement, education, and collaboration to inspire positive changes in environmental stewardship.

**Key Objectives and Features:**

1. **User Authentication:**
   * EcoHub provides a seamless user authentication process, allowing users to sign up and log in using their Google accounts. This streamlined approach ensures easy access to platform features while maintaining security and user identity management.
2. **User Profiles:**
   * Each user has a dedicated profile page where they can manage their account information. This includes details such as their EcoCoin balance (a virtual currency used for eco-friendly transactions within the platform), order history, waste collection requests status, and more. The user profile serves as a central hub for personalized experiences and interactions on EcoHub.
3. **Waste Collection Requests:**
   * One of the core functionalities of EcoHub is its waste collection request system. Users can initiate waste collection requests by filling out a simple form specifying the type of waste (e.g., recyclables, e-waste, organic waste) and its location. These requests are then routed to waste management partners who handle the collection and disposal process efficiently.
4. **E-commerce Platform:**
   * EcoHub hosts an e-commerce platform offering a diverse range of eco-friendly products and services. Users can browse and purchase items such as organic foods, sustainable fashion apparel, reusable household goods, eco-friendly gadgets, and more. This platform not only promotes sustainable consumerism but also supports businesses that prioritize environmental responsibility.
5. **Admin Dashboard:**
   * Administrators have access to a powerful dashboard equipped with tools for managing various aspects of the platform. The admin dashboard allows administrators to:
     + Manage user accounts, permissions, and roles.
     + Monitor and track waste collection requests, including their status and processing.
     + Analyze platform metrics such as user engagement, product sales, waste management effectiveness, and other key performance indicators (KPIs).
     + Oversee and update product listings, ensuring a curated selection of sustainable products for users.

By combining these features, EcoHub aims to create a vibrant and interconnected ecosystem where individuals, businesses, waste management agencies, and environmental advocates can collaborate towards a common goal of sustainability. The platform's educational resources, interactive features, and community-driven initiatives empower users to make informed choices and contribute positively to environmental conservation efforts.

**2 System Study**

**2.1 Existing System**

In the absence of EcoHub, individuals interested in adopting sustainable living practices often face several challenges:

1. **Fragmented Resources:**

Information and resources related to sustainable living are scattered across various platforms, making it difficult for users to find comprehensive guidance and support.

1. **Limited Access to Eco-friendly Products:**

Finding eco-friendly products and services often requires extensive research, and options may be limited depending on geographic location and availability.

1. **Lack of Community Support:**

Without a dedicated platform for sustainability enthusiasts, individuals may feel isolated and lack opportunities for collaboration, knowledge sharing, and community engagement.

1. **Inefficient Waste Management:**

Waste management processes may be inefficient or inaccessible to users, leading to improper disposal practices and environmental harm.

1. **Complexity in Monitoring Progress:**

Tracking individual progress towards sustainability goals can be challenging without centralized tools and mechanisms for monitoring and feedback.

**2.2 Proposed System**

EcoHub addresses these shortcomings by offering a comprehensive and user-friendly platform designed to promote sustainable living practices effectively. Key features of the proposed system include:

1. **Centralized Information Hub:**

EcoHub serves as a centralized repository of resources, tips, articles, and guides on various aspects of sustainable living, making it easier for users to access relevant information.

1. **E-commerce Marketplace:**

The platform hosts an e-commerce marketplace where users can discover and purchase eco-friendly products and services, ranging from organic foods to sustainable fashion items, all vetted for their environmental impact.

1. **Waste Collection Services:**

EcoHub facilitates waste collection services by allowing users to submit requests for pickup, enabling efficient and environmentally responsible disposal of waste materials.

1. **Community Engagement:**

Through forums, discussion boards, and collaboration tools, EcoHub fosters a vibrant community of sustainability enthusiasts who can share ideas, collaborate on projects, and support each other in adopting eco-conscious behaviors.

1. **User Profiles and Progress Tracking:**

Each user has a personalized profile where they can track their sustainability journey, view their EcoCoin balance, monitor waste collection requests, and track their order history.

1. **Administrator Dashboard:**

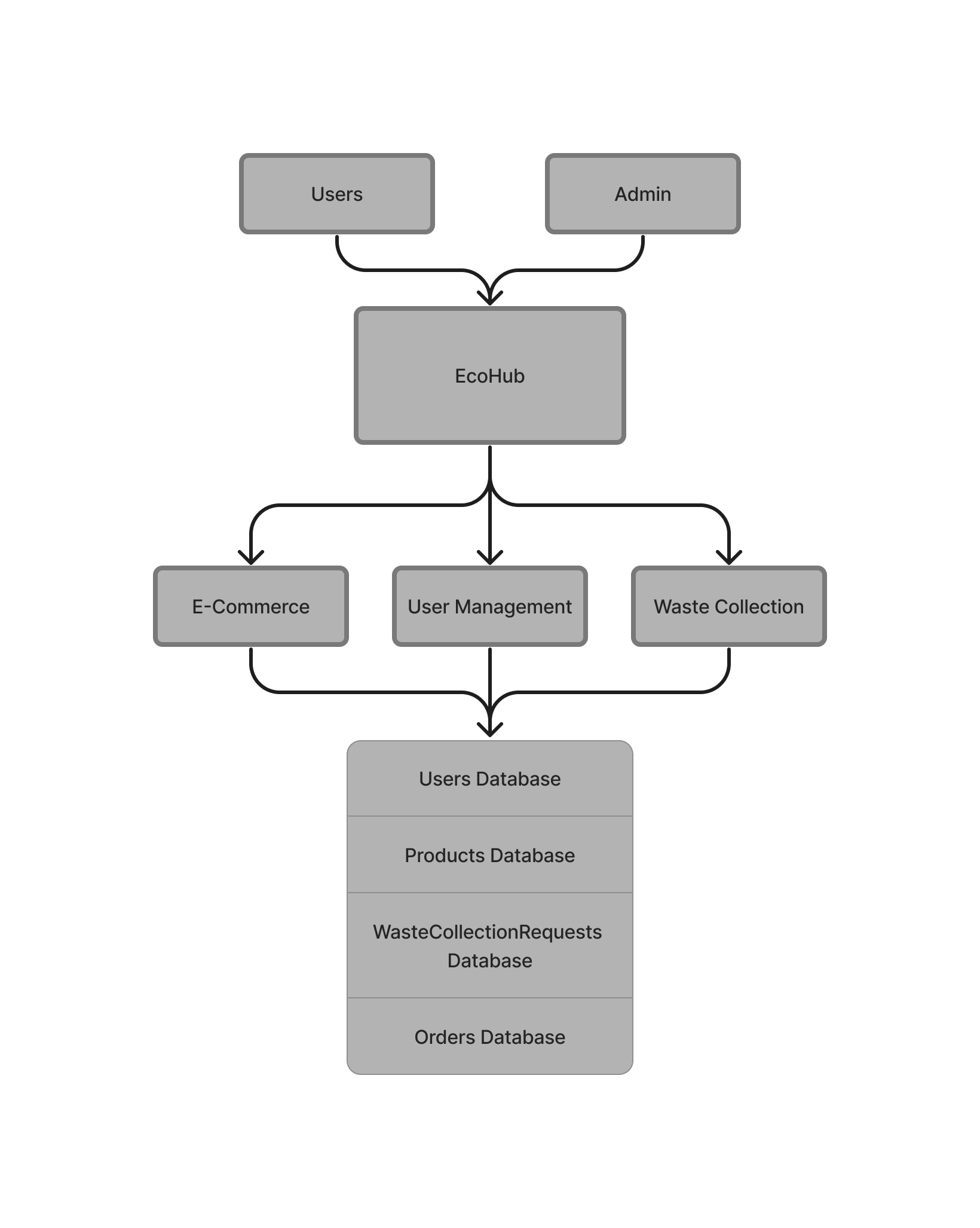
Administrators have access to a dedicated dashboard for managing user accounts, monitoring platform activity, analyzing data trends, and overseeing product listings and waste collection operations.

By providing a unified platform that combines education, commerce, and community engagement, EcoHub aims to empower individuals to make informed choices and take meaningful actions towards a more sustainable future.Top of Form

**2.3 Data Flow Diagram**

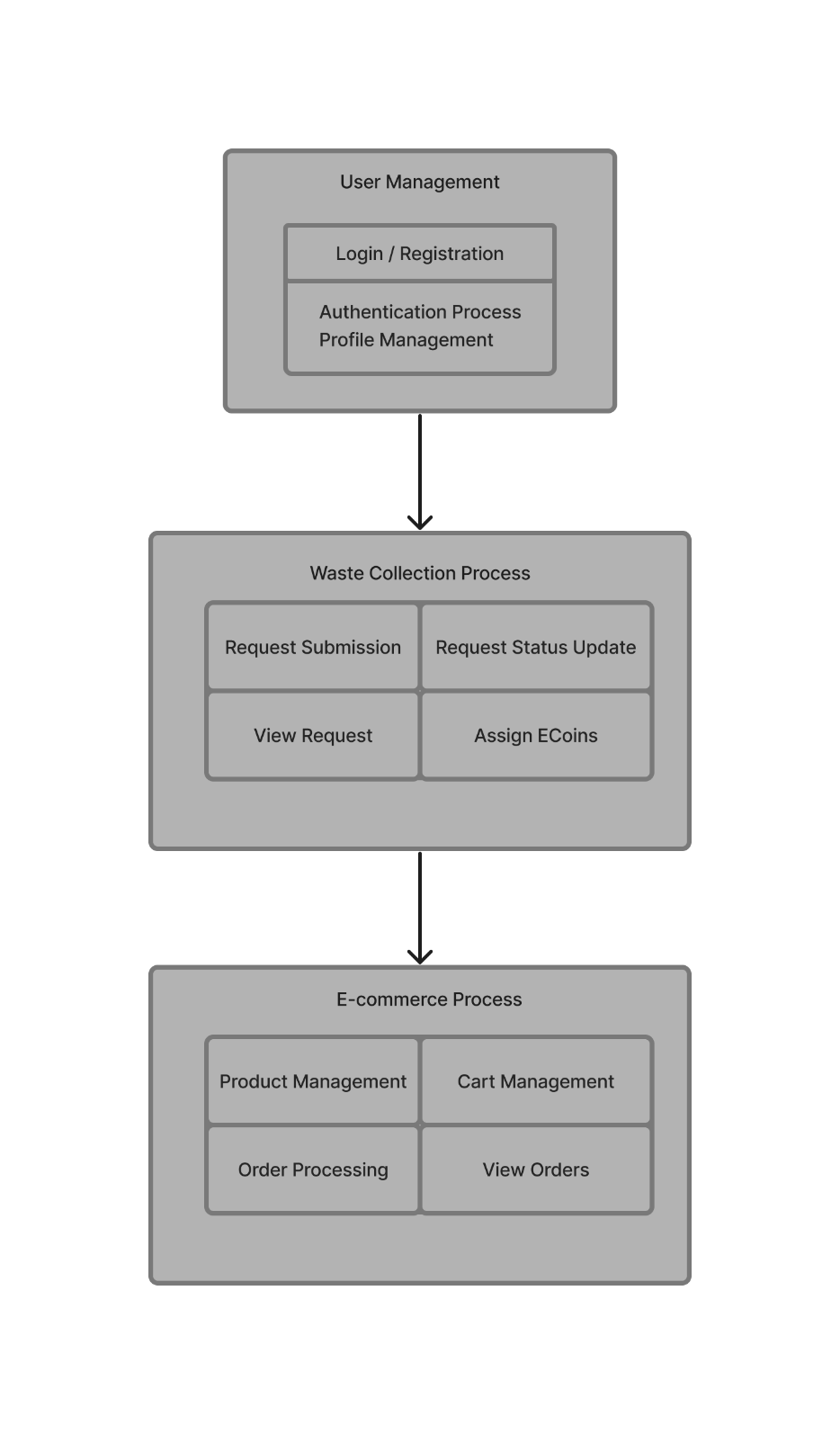
**Level 0**

The Level 0 DFD provides an overview of the entire system, showing major processes and their interactions with external entities.



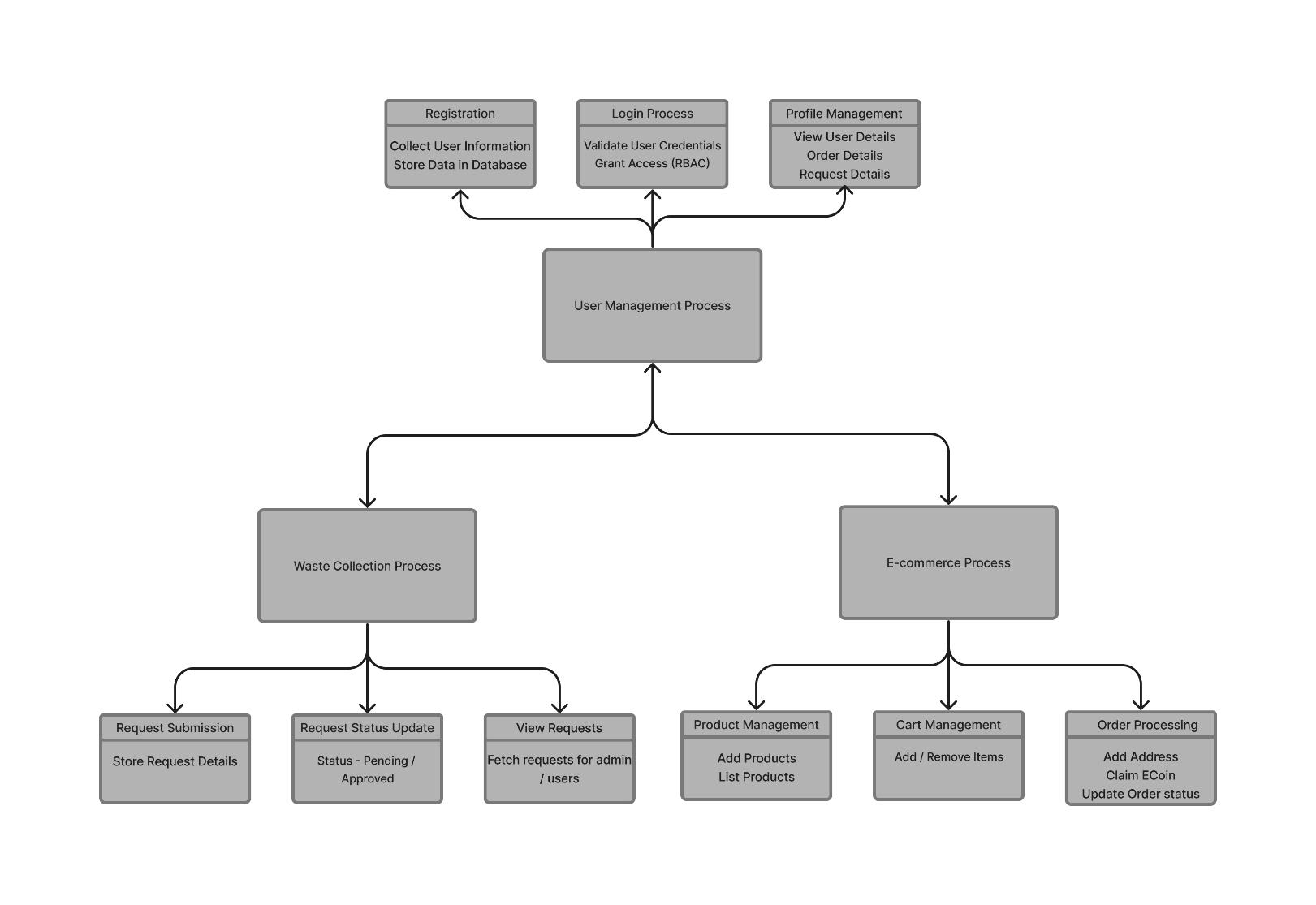
**Level 1**

The Level 1 DFD expands on the Level 0 processes, breaking them down into more detailed subprocesses.



**Level 2**

The Level 2 DFD delves deeper into specific subprocesses, detailing the data transformations and interactions within each subprocess.



**2.4 ER Diagram**

An Entity-Relationship (ER) diagram is a visual representation of the entities, attributes, relationships, and constraints within a database schema.

A diagram of a flowchart

Description automatically generated

1. **System Configuration**

System configuration involves the setup and management of various components within a computer system or IT environment. It encompasses hardware, software, network settings, security configurations, and more. Proper system configuration is essential for ensuring that systems operate efficiently, securely, and in line with organizational requirements.

**3.1 Hardware Configuration**

The hardware configuration outlined below provides the minimum requirements necessary to run the EcoHub platform effectively:

1. **Server Infrastructure:**
   * **CPU:** A dual-core processor with a clock speed of 1.8 GHz or higher is recommended. This ensures sufficient processing power to handle server-side operations efficiently.
   * **RAM:** At least 4 GB of DDR4 RAM is advised for smooth operation. Sufficient RAM helps in handling concurrent user requests, database operations, and background tasks.
   * **Storage:** A minimum of 50 GB of storage space, whether SSD or HDD, is needed for storing the operating system, application files, and data generated by the platform.
   * **Network:** A 1 Gbps Ethernet interface ensures fast network connectivity, which is crucial for handling data transmission between the server and client devices.
2. **Networking Equipment:**
   * **Router:** A basic router with NAT capability is required to manage network traffic and provide internet connectivity to the server and client devices.
   * **Switch:** A Gigabit Ethernet switch facilitates high-speed local network connectivity, ensuring smooth communication between devices within the network.
   * **Firewall:** Implementing a basic firewall, either as a hardware appliance or software solution, enhances network security by monitoring and controlling incoming and outgoing traffic.
3. **Client Devices:**
   * **Desktop/Laptop:**
     + **CPU:** Similar to the server, a dual-core processor with a clock speed of 1.5 GHz or higher is suitable for client devices.
     + **RAM:** 2 GB of DDR3 RAM is recommended for desktops or laptops to ensure adequate memory for running web browsers and accessing the EcoHub platform smoothly.
     + **Storage:** Around 100 MB of available disk space is sufficient for storing browser cache, temporary files, and downloaded content related to the EcoHub platform.
     + **Network:** Client devices should have Ethernet or Wi-Fi connectivity options to connect to the internet and access the EcoHub platform.
   * **Mobile/Tablet:**
     + **Browser Compatibility:** Client devices should support modern web browsers such as Google Chrome, Mozilla Firefox, or Safari to access the EcoHub web application.
     + **Operating Systems:**
       - For Android devices, the platform should be compatible with Android OS version 5.0 (Lollipop) or higher.
       - For iOS devices, compatibility should extend to iOS version 10 or higher to ensure optimal performance and functionality.
4. **Peripheral Devices:**
   * EcoHub may interact with optional peripheral devices like printers, scanners, or cameras based on user requirements. These peripherals should have standard USB or wireless connectivity protocols compatible with the client devices.

This hardware configuration ensures that both the server infrastructure and client devices meet the minimum requirements to run the EcoHub platform efficiently, providing users with a seamless experience while accessing the platform's features and functionalities.

**3.2 Software Configuration**

The software configuration of the EcoHub platform encompasses various components and technologies carefully selected to ensure a robust, secure, and scalable application ecosystem. Let's delve deeper into each aspect of the software configuration:

1. **Frontend:**
   * **Framework:** The frontend of EcoHub is built using HTML, CSS3, and JavaScript with the React.js library. React.js facilitates the development of dynamic and interactive user interfaces, enabling smooth navigation and responsive design.
   * **Authentication:** GoogleAuth is utilized for user authentication, leveraging OAuth mechanisms to authenticate users securely and manage user sessions.
2. **Backend:**
   * **Framework:** The backend infrastructure is powered by Node.js, a server-side JavaScript runtime environment. Node.js enables efficient handling of backend operations, such as API integrations, business logic implementation, and database interactions.
3. **Database:**
   * **Type:** The database management system used is Google Firebase, specifically Firestore. Firestore is a NoSQL cloud database that offers real-time data synchronization, scalability, and offline support, making it ideal for applications like EcoHub that require flexible and scalable data storage.
4. **Security:**
   * **Encryption:** To ensure data security, encryption mechanisms are applied, especially for sensitive data such as user credentials, payment information, and personal details. Encryption protocols like HTTPS and data encryption standards are implemented to protect data both in transit and at rest.
5. **Development:**
   * **Methodology:** EcoHub follows an Agile development methodology, emphasizing iterative development, collaboration, and continuous improvement. Agile practices allow for quick adaptation to changing requirements, faster development cycles, and better stakeholder engagement.
   * **Version Control:** Git is used for version control and collaborative development. Git enables team members to work concurrently on the codebase, manage code versions, track changes, and facilitate code review processes.

The software configuration of EcoHub is designed with modularity and scalability in mind, allowing seamless integration of new features, enhancements, and third-party services. It prioritizes security and data integrity through encryption practices and utilizes modern technologies to deliver a user-friendly, efficient, and reliable platform for users, administrators, and stakeholders.

**4 Details of Software**

In the EcoHub project, various software components are utilized to enable its functionality. Below is a breakdown of the software used along with their roles in the system:

**4.1 Overview of Front End**

The frontend of the EcoHub platform serves as the user-facing interface, providing an intuitive and engaging experience for users as they interact with various features and functionalities. Developed primarily using React.js, a powerful JavaScript library for building user interfaces, the frontend encompasses several key components and technologies aimed at delivering a seamless user experience. Below is an overview of the frontend architecture and its core elements:

**4.1.1 React.js:**

React.js forms the foundation of the EcoHub frontend, offering a component-based approach to building interactive web applications. Leveraging React's declarative and efficient nature, the frontend components are structured in a modular and reusable manner, promoting code maintainability and scalability. React's virtual DOM efficiently updates the UI in response to user interactions and data changes, ensuring optimal performance across different devices and browsers.

**4.1.2 React Router:**

React Router is utilized for client-side routing within the EcoHub application, enabling navigation between different pages and components without the need for full-page reloads. By defining routes and associated components, React Router facilitates seamless navigation flows, allowing users to explore different sections of the platform while maintaining a single-page application experience.

**4.1.3 Firebase Authentication:**

Firebase Authentication is integrated into the frontend to manage user authentication and authorization. Through Firebase's robust authentication services, users can securely sign in to the EcoHub platform using their Google accounts. Firebase Authentication handles user sessions, authentication states, and user roles, ensuring a smooth and secure login experience for users.

**4.1.4 Firebase Firestore:**

Firestore, a NoSQL cloud database provided by Firebase, serves as the primary data storage solution for the EcoHub frontend. Through Firestore, user profiles, product information, orders, and waste collection requests are stored and managed in real-time. The frontend interacts with Firestore to fetch and update data dynamically, enabling features such as real-time updates and synchronization across multiple devices.

**4.1.5 User Interface Design:**

The EcoHub frontend is designed with a focus on usability, aesthetics, and responsiveness. Custom CSS stylesheets are employed to define the layout, typography, colors, and visual elements of the user interface, ensuring a cohesive and visually appealing design language. Icons from FontAwesome are incorporated to enhance the user experience and provide intuitive navigation cues.

**4.1.6 Dynamic Content Rendering:**

Dynamic content rendering is a key aspect of the EcoHub frontend, allowing users to view and interact with personalized content based on their preferences and actions. Through React's state management and lifecycle methods, the frontend dynamically renders content such as user profiles, product listings, order details, and waste collection requests, ensuring that users receive relevant and up-to-date information.

By combining these frontend technologies and design principles, the EcoHub platform delivers an immersive and user-centric experience, empowering users to contribute to environmental sustainability while enjoying a seamless and intuitive web application interface.

**4.2 Overview of Back End**

* **Node.js:**

The backend server for the EcoHub platform is built using Node.js, a JavaScript runtime environment. Node.js enables server-side scripting and provides an event-driven architecture, facilitating the handling of concurrent connections and asynchronous operations.

* **Express.js:**

Express.js is used as the web application framework for Node.js, providing a robust set of features for building web servers and APIs. Express simplifies routing, middleware integration, and request handling.

* **Firebase Admin SDK:**

The Firebase Admin SDK is utilized to interact with Firebase services from the backend server. It enables server-side functionalities such as authentication, database access, and cloud messaging.

* **Google Cloud Functions:**

Google Cloud Functions are used to deploy serverless functions that execute in response to events triggered by Firebase services. These functions handle background tasks such as sending notifications, processing data, and performing server-side logic.

**4.3 Additional Tools and Libraries:**

* **FontAwesome:**

FontAwesome is utilized to access an extensive collection of icons, enhancing the visual appeal and user interface elements throughout the EcoHub platform. Icons are strategically placed to improve user experience and navigation.

* **Custom CSS Stylesheets:**

Custom CSS stylesheets play a crucial role in defining the layout, design, and visual elements of the EcoHub web application. These stylesheets ensure consistency in the user interface design, providing a visually appealing and intuitive experience for users.

* Moreover, for analytics and data visualization purposes, the EcoHub platform integrates the following tool:
* **React ChartJS 2 (Bar Chart):**

Utilizing the **Bar** component from 'react-chartjs-2', EcoHub generates analytical insights such as the most sold products. This integration enables the platform to present data in visually informative ways, aiding users and administrators in understanding trends and making data-driven decisions.

By leveraging these software components and tools, EcoHub delivers a comprehensive and engaging user experience while effectively addressing environmental and sustainability challenges.

**5 System Design**

The system design of the EcoHub platform encompasses the architectural framework, database schema, and interaction flow between various components. It aims to provide a comprehensive overview of how different modules and functionalities are organized and interconnected within the application. Below are the key aspects of the system:

**5.1 Architectural Design**

The EcoHub platform follows a client-server architecture, where the frontend client interacts with backend server infrastructure to access and manipulate data. The frontend client, developed using React.js, communicates with the backend server directly via SDKs APIs to perform operations such as user authentication, data retrieval, and.

**Frontend Client**

The frontend of the EcoHub platform, built using React.js, forms the user-facing part of the application. Here are the key aspects of the frontend architecture:

* **Component-Based Structure:** Utilizes React.js for a modular and component-based UI development approach. Each UI element is encapsulated within reusable components.
* **State Management:** Leverages React's state and props mechanism for managing component state and passing data between components.
* **Virtual DOM:** React optimizes rendering through a virtual DOM, enhancing performance by updating only the necessary parts of the UI.
* **Routing:** Uses React Router for client-side routing, enabling navigation between different views without full page reloads.

**Backend**

The backend of the EcoHub platform provides the necessary data and logic support to the frontend. Here are the simplified components of the backend:

* **Serverless Backend:** Utilizes serverless architecture (e.g., Firebase Functions) or a simplified backend service (e.g., Firebase Firestore) without the need for a separate backend server.
* **Database:** Stores and retrieves data using a NoSQL database like Firebase Firestore or a similar cloud-based database service.
* **Client-Server Communication:** Frontend communicates directly with the backend database or serverless functions through client-side SDKs or APIs provided by the chosen backend service.

**Communication Between Frontend and Backend**

* **Direct Communication:** The frontend communicates directly with the backend services or database using client-side SDKs or APIs.
* **Data Retrieval:** Frontend retrieves data (such as user profiles, product information, etc.) from the backend as needed for display and functionality.
* **User Interaction:** Backend processes user input and requests (e.g., form submissions, user authentication) and returns relevant data or responses to the frontend.
* **Authentication and Authorization:** User authentication and authorization may be simplified using client-side authentication mechanisms provided by the chosen backend service, avoiding the need for JWT or complex server-side authentication.

**5.2 Input Design**

The input design of the EcoHub platform focuses on providing user-friendly interfaces for users to input data and interact with the system effectively. Various forms, fields, and controls are designed to collect user inputs and initiate actions. Here are some key aspects of the input design:

* **User Registration/Login Form:** Users can register or log in to the platform using their Google account credentials. The login form prompts users to enter their email and password for authentication.
* **User Profile Management Form:** Authenticated users can view their profile information, including display name, email. The profile management form includes input fields for users to view their details.
* **Shopping Cart:** When users add products to their shopping cart. Input fields within the shopping cart interface allow users to adjust the quantity or remove items from their cart.
* **Waste Collection Request Form:** Users can submit waste collection requests by providing details such as waste type and location. The request form includes input fields for users to enter the required.

**5.3 Output Design**

The output design of the EcoHub platform focuses on presenting information and feedback to users in a clear and meaningful manner. Various components, layouts, and formats are employed to display data, notifications, and status updates. Here are some key aspects of the output design:

* **User Profile Display:** After successful login, users are greeted with a personalized welcome message and their profile information, such as display name and email. The user profile page presents this information in a visually appealing layout.
* **Product Listings:** Products are displayed in a grid or list format, showcasing key details such as product name, price, and image. Each product listing includes a thumbnail image and a brief description to provide users with relevant information.
* **Order Confirmation:** When users place an order, they receive a confirmation message indicating the successful completion of the transaction. Order details, including order ID and total amount, are displayed for reference.
* **Waste Collection Request Acknowledgment:** After submitting a waste collection request, users receive a confirmation message confirming the receipt of their request. The acknowledgment message provides users with a unique request ID and informs them that their request is being processed.
* **Error Messages and Alerts:** In case of errors or validation issues, appropriate error messages and alerts are displayed to users to notify them of the problem. Error messages are designed to be informative and guide users on how to rectify the issue.

**5.4 Database Design**

The database schema of the EcoHub platform plays a crucial role in efficiently managing various types of data and supporting the platform's functionalities. Let's extend the content to provide a more detailed overview of each collection within Firebase Firestore:

**Users Collection**

The **Users** collection is designed to store user profiles and relevant information:

* **Fields:**
  + **userID:** Unique identifier for each user.
  + **email:** User's email address, used for authentication and communication.
  + **displayName:** User's display name.
  + **role:** Indicates the user's role or privileges within the platform (., 'user' or 'admin').
  + **ECoin balance:** Tracks the virtual currency balance associated with each user for transactions and rewards.

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraint Name** |
| userID | string | Primary Key |
| email | string |  |
| displayName | string |  |
| role | string |  |
| Ecoin | number |  |

**Products Collection**

The **Products** collection contains details about the products available for users to browse and purchase:

* **Fields:**
  + **productID:** Unique identifier for each product.
  + **name:** Name or title of the product.
  + **description:** Brief description or details about the product.
  + **price:** Price of the product, typically in the platform's currency (e.g., dollars, credits).
  + **imageURL:** URL pointing to the product's image or thumbnail for visual representation.

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraint Name** |
| productID | string | Primary Key |
| name | string | Candidate Key |
| description | string |  |
| price | number |  |
| imageURL | string |  |

**Orders Collection**

The **Orders** collection tracks user transactions and order history:

* **Fields:**
  + **orderID: string (Primary Key) -** Unique identifier for each order placed.
  + **userID: string (Foreign Key) -** Identifies the user who placed the order, linking back to the Users collection.
  + **timestamp: timestamp -** Date and time when the order was placed, useful for chronological sorting and analysis.
  + **items: string -** Nested array containing details of each item in the order, including product ID, name, quantity, and subtotal price per item.
  + **name: string -** Name of the user placing the order.
  + **address: string -** Address where the order is to be delivered.
  + **amount: decimal -** Total amount of the order.

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraint Name** |
| orderID | string | Primary Key |
| userID | string | Foreign Key |
| name | string |  |
| address | string |  |
| amount | price |  |
| items | string |  |

**Waste Collection Requests Collection**

The **Waste Collection Requests** collection manages requests for waste collection services initiated by users:

* **Fields:**
  + **reqID:** Unique identifier for each waste collection request.
  + **userID:** Identifies the user who made the request, linking back to the Users collection.
  + **wasteType:** Type or category of waste for the collection (e.g., recyclables, electronic waste).
  + **location:** Location details where the waste collection is requested (e.g., address, coordinates).
  + **status:** Tracks the status of the request (e.g., 'pending', 'approved', 'completed').

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Datatype** | **Constraint Name** |
| reqID | string | Primary Key |
| userID | string | Foreign Key |
| wasteType | string |  |
| location | number |  |
| status | string |  |

By structuring data into these collections and defining relevant fields, the EcoHub platform can efficiently manage user profiles, product catalog, orders, and waste collection requests, providing a seamless experience for users and administrators while facilitating data-driven operations and analytics within the platform.

**6 Source Code**

**Server.js**

const express = require('express');

const cors = require('cors');

const admin = require('firebase-admin');

const app = express();

const port = process.env.PORT || 5000;

app.use(cors());

app.use(express.json()); *// Parse JSON requests*

const serviceAccount = require('./ecohub-2a3f4-firebase-adminsdk-kz1ma-8f25a4bc5a.json');

admin.initializeApp({

  credential: admin.credential.cert(serviceAccount),

});

const firestore = admin.firestore();

*// Route for handling waste collection requests*

app.post('/api/waste-collection', async (*req*, *res*) => {

  try {

    const { user, wasteDetails } = req.body;

*// Store waste collection request in Firestore*

    const collectionRef = firestore.collection('wasteCollectionRequests');

    await collectionRef.add({

      userId: user.uid,

      wasteDetails,

      timestamp: admin.firestore.FieldValue.serverTimestamp(),

    });

    res.status(200).json({ message: 'Waste collection request received.' });

  } catch (error) {

    console.error(error);

    res.status(500).json({ error: 'Internal server error' });

  }

});

app.listen(port, () => {

  console.log(`Server is running on port ${port}`);

});

**AdminDashboard.js**

import React, { useEffect, useState } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { auth, firestore } from '../../firebase';

import { doc, getDoc, collection } from 'firebase/firestore';

*// import { Link } from 'react-router-dom';*

const checkAdminStatus = async (*currentUserUid*) => {

  try {

    console.log('Checking admin status for user:', currentUserUid);

    const usersRef = collection(firestore, 'users');

    const userDoc = await getDoc(doc(usersRef, currentUserUid));

    console.log('User document:', userDoc.data());

    return userDoc.exists() && userDoc.data().role === 'admin';

  } catch (error) {

    console.error('Error checking admin status:', error);

    return false;

  }

};

const AdminDashboard = () => {

  const [isAdmin, setIsAdmin] = useState(false);

  const navigate = useNavigate();

  useEffect(() => {

    const checkAdmin = async () => {

      try {

        const currentUser = auth.currentUser;

        const currentUserUid = currentUser ? currentUser.uid : null;

        if (!currentUserUid) {

          console.error('User not authenticated');

          navigate('/'); *// Redirect to the home page or login page*

          return;

        }

        const isAdminUser = await checkAdminStatus(currentUserUid);

        setIsAdmin(isAdminUser);

        if (!isAdminUser) {

*// Redirect or handle unauthorized access*

          console.log('Unauthorized access to admin dashboard');

*// Optionally, redirect the user to another page*

          navigate('/');

        }

      } catch (error) {

        console.error('Error checking admin status:', error);

      }

    };

    checkAdmin();

  }, [navigate]);

  return (

    <div>

      <h2>Admin Dashboard</h2>

      <nav>

        <ul>

          <li><Link to="/admin/requests">Waste Collection Requests</Link></li>

          <li><Link to="/admin/users">Users</Link></li>

          <Link to="/admin/order-details">Order Details</Link>

        </ul>

      </nav>

    </div>

  );

};

export default AdminDashboard;

**AdminRequest.js**

*// src/components/Admin/AdminRequests.js*

import React, { useEffect, useState } from 'react';

import { collection, onSnapshot, doc, deleteDoc, updateDoc, increment } from 'firebase/firestore';

import { firestore } from '../../firebase';

import { getFirestore } from 'firebase/firestore';

const db = getFirestore();

const AdminRequests = () => {

  const [requests, setRequests] = useState([]);

  useEffect(() => { *// Fetch waste collection requests data*

    const requestsRef = collection(firestore, 'wasteCollectionRequests');

    const requestsUnsubscribe = onSnapshot(requestsRef, (*snapshot*) => {

      const requestsData = snapshot.docs.map((*doc*) => ({ id: doc.id, ...doc.data() }));

      setRequests(requestsData);

    });

    return () => {*// Unsubscribe from snapshots to avoid memory leaks*

      requestsUnsubscribe();

    };

  }, []);

  const handleApproveRequest = async (*request*) => {

    try {*// Update the status of the request*

      const requestDocRef = doc(firestore, 'wasteCollectionRequests', request.id);

      await updateDoc(requestDocRef, { status: 'Approved', wasteType: request.wasteType, location: request.location });

*// Increment the ECoin of the user*

      const userDocRef = doc(firestore, 'users', request.userId);

      await updateDoc(userDocRef, { ECoin: increment(10) });

      console.log('Request approved successfully');

    } catch (error) {

      console.error('Error approving request:', error);

    }

  };

  const handleRejectRequest = async (*request*) => {

    try { *// Update the status of the request to "Rejected"*

      const requestDocRef = doc(db, 'wasteCollectionRequests', request.id);

      await updateDoc(requestDocRef, { status: 'Rejected' });

*// Optionally, you can add code here to handle other actions when a request is rejected*

    } catch (error) {

      console.error('Error during waste collection request rejection:', error);

    }

  };

  const handleDeleteRequest = async (*requestId*) => {

    try {

      const requestDocRef = doc(firestore, 'wasteCollectionRequests', requestId);

      await deleteDoc(requestDocRef);

      console.log('Request deleted successfully');

    } catch (error) {

      console.error('Error deleting request:', error);

    }

  };

  return (

    <div>

      <h2>Waste Collection Requests</h2>

      <table className='userDataDiv'>

        <thead>

          <tr>

            <th>Request ID</th>

            <th>User ID</th>

            <th>Waste Type</th>

            <th>Location</th>

            <th>Status</th>

            <th>Actions</th>

          </tr>

        </thead>

        <tbody>

          {requests.map((*request*) => (

            <tr key={request.id}>

              <td>{request.id}</td>

              <td>{request.userId}</td>

              <td>{request.wasteType}</td>

              <td>{request.location}</td>

              <td>{request.status}</td>

              <td>

                <button onClick={() => handleApproveRequest(request)}>Approve</button>

                <button onClick={() => handleRejectRequest(request)}>Reject</button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </div>

  );

};

export default AdminRequests;

**7 Testing**

Testing is an essential phase in the development process of the EcoHub platform to ensure its functionality, reliability, and performance meet the intended requirements. The testing process involves various types of testing methodologies to identify and resolve any defects or issues before deployment. Here is an overview of the testing conducted for the EcoHub platform:

**7.1 Unit Testing:**

Unit testing focuses on testing individual units or components of the application in isolation. Key aspects of unit testing for EcoHub include:

* **Isolation Testing:**

Each function, method, or module is tested independently to verify its functionality.

* **Test Cases:**

Mock data and test cases are designed to simulate different scenarios and edge cases, ensuring comprehensive coverage.

* **Tools:**

Jest and React Testing Library are used for unit testing React components, Redux actions, and utility functions.

**Test Case:**

1. **Scenario:** Testing a Redux action to add an item to the cart.
   * **Test Case:** Dispatch the add-to-cart action with mock item data.
   * **Expected Outcome:** Redux store should update with the added item in the cart.
   * **Actual Outcome:** The Redux store is updated as expected.
2. **Scenario:** Testing a utility function to format currency.
   * **Test Case:** Call the formatCurrency function with a mock currency value.
   * **Expected Outcome:** The currency value should be formatted correctly (e.g., $100.00).
   * **Actual Outcome:** The function returns the correctly formatted currency value.

**7.2 Integration Testing:**

Integration testing evaluates the interactions and interfaces between various modules or components of the EcoHub platform. Key aspects of integration testing include:

* **Component Interaction:**

Testing how different components interact and exchange data within the application.

* **API Integration:**

Verifying the integration of backend APIs with frontend components to ensure data consistency and accuracy.

**Test Case:**

1. **Scenario:** Testing the interaction between the shopping cart component and checkout process.
   * **Test Case:** Add items to the cart and proceed to checkout.
   * **Expected Outcome:** Cart items should be displayed in the checkout process.
   * **Actual Outcome: Cart items are successfully displayed during checkout.**
2. **Scenario:** Testing API integration for user authentication.
   * **Test Case:** Submit valid user credentials to the login API endpoint.
   * **Expected Outcome:** Receive a success response and authenticate the user.
   * **Actual Outcome:** User authentication is successful, and the API returns an authentication token.

**7.3 User Acceptance Testing (UAT):**

User Acceptance Testing (UAT) is crucial for validating the EcoHub platform against user requirements and expectations. Key aspects of UAT include:

* **Real User Testing:**

Involving actual users or stakeholders to test the application's usability and functionality.

* **Feedback Collection:**

Gathering feedback and insights from users to identify areas for improvement and optimization.

* **Usability Testing:**

Evaluating the overall user experience, navigation, and accessibility of the platform.

**Test Case:**

1. **Scenario:** Real user testing for registering a new account.
   * **Test Case:** Have a user register a new account with valid information.
   * **Expected Outcome:** The user should be able to register successfully and receive a confirmation.
   * **Actual Outcome:** The user successfully registers, and a confirmation email is sent.
2. **Scenario:** Usability testing for navigating product categories.
   * **Test Case:** Ask users to find a specific product category using the navigation menu.
   * **Expected Outcome:** Users should locate the desired category within a reasonable time.
   * **Actual Outcome:** Most users find the category easily, but a few struggle with the navigation.

**7.4 Performance Testing:**

Performance testing assesses the responsiveness, stability, and scalability of the EcoHub platform under various conditions. Key aspects of performance testing include:

* **Load Testing:**

Evaluating system performance under expected and peak load conditions to ensure responsiveness.

* **Stress Testing:**

Testing system limits and identifying breaking points under extreme load scenarios.

* **Scalability Testing:**

Assessing the platform's ability to handle increasing loads and scaling resources efficiently.

**Test Case:**

1. **Scenario:** Load testing the product search functionality.
   * **Test Case:** Simulate multiple users searching for products simultaneously.
   * **Expected Outcome:** The search functionality should handle the load without significant latency.
   * **Actual Outcome:** The search functionality remains responsive under load, with acceptable response times.
2. **Scenario:** Stress testing the checkout process with a high number of simultaneous transactions.
   * **Test Case:** Simulate a large number of users checking out with different items.
   * **Expected Outcome:** The checkout process should handle the load without crashing or timeouts.
   * **Actual Outcome:** The checkout process remains stable, but some timeouts occur under extreme load.

**7.5 Security Testing:**

Security testing is paramount to identify and mitigate potential vulnerabilities and threats within the EcoHub platform. Key aspects of security testing include:

* **Vulnerability Assessment:**

Identifying and addressing vulnerabilities such as SQL injection, cross-site scripting (XSS), and authentication flaws.

* **Penetration Testing:**

Simulating real-world attacks to identify weaknesses and strengthen security measures.

* **Compliance Testing:**

Ensuring the platform complies with security standards and regulations.

**Test Case:**

1. **Scenario:** Vulnerability assessment for input fields.
   * **Test Case:** Attempt SQL injection in the search input field.
   * **Expected Outcome:** The system should detect and prevent SQL injection attempts.
   * **Actual Outcome:** The system properly sanitizes input and prevents SQL injection vulnerabilities.
2. **Scenario:** Penetration testing for unauthorized access attempts.
   * **Test Case:** Attempt to access restricted admin functionalities without proper credentials.
   * **Expected Outcome:** The system should block unauthorized access attempts and log them.
   * **Actual Outcome:** Unauthorized access attempts are blocked, and logs show security violations.

**7.6 Regression Testing:**

Regression testing ensures that new updates or changes do not introduce defects or regressions into the EcoHub platform. Key aspects of regression testing include:

* **Automated Testing:**

Utilizing automated test suites to perform regression tests efficiently and validate existing functionalities.

* **Version Control:**

Testing across different versions and branches to maintain consistency and stability.

**Test Case:**

1. **Scenario:** Testing cart functionality after a code update.
   * **Test Case:** Add items to the cart and proceed to checkout after a recent code deployment.
   * **Expected Outcome:** Cart functionality should remain intact without regression issues.
   * **Actual Outcome:** Cart functionality works as expected without any regression issues.
2. **Scenario:** Testing user authentication across different versions.
   * **Test Case:** Verify user login functionality across old and new versions of the platform.
   * **Expected Outcome:** User authentication should work consistently across different versions.
   * **Actual Outcome:** User authentication remains consistent across versions, confirming no regression in this area.

**8 Implementation**

The implementation phase of the EcoHub project involved translating the design and requirements into a fully functional application. This phase included the development of both the frontend and backend components, integration of third-party services, database setup, and deployment of the application. Here's an overview of the implementation process:

**Frontend Development:**

* The frontend of the EcoHub platform was developed using React.js, a popular JavaScript library for building user interfaces.
* UI/UX designs were translated into responsive and interactive web pages using HTML, CSS, and JavaScript.
* Components such as the user dashboard, waste collection form, e-commerce pages, and admin interface were implemented to provide different functionalities to users.
* React Router was used for client-side routing to enable navigation between different pages of the application.
* Firebase Authentication was integrated to handle user authentication and authorization, allowing users to log in securely using their Google accounts.
* Various third-party libraries and frameworks, such as Material-UI for UI components and Axios for HTTP requests, were used to enhance the development process and improve the user experience.

**Backend Development:**

* The backend of the EcoHub platform was built using Firebase, a comprehensive platform provided by Google for developing web and mobile applications.
* Firebase Firestore, a NoSQL cloud database, was used to store and manage user data, product information, waste collection requests, and other application data.
* Firebase Authentication was utilized to authenticate users and secure access to the application's resources.
* Cloud Functions for Firebase were implemented to handle server-side logic, such as processing waste collection requests, sending email notifications, and performing other backend tasks.
* Firebase Storage was used to store and manage user-uploaded files, such as product images and profile.

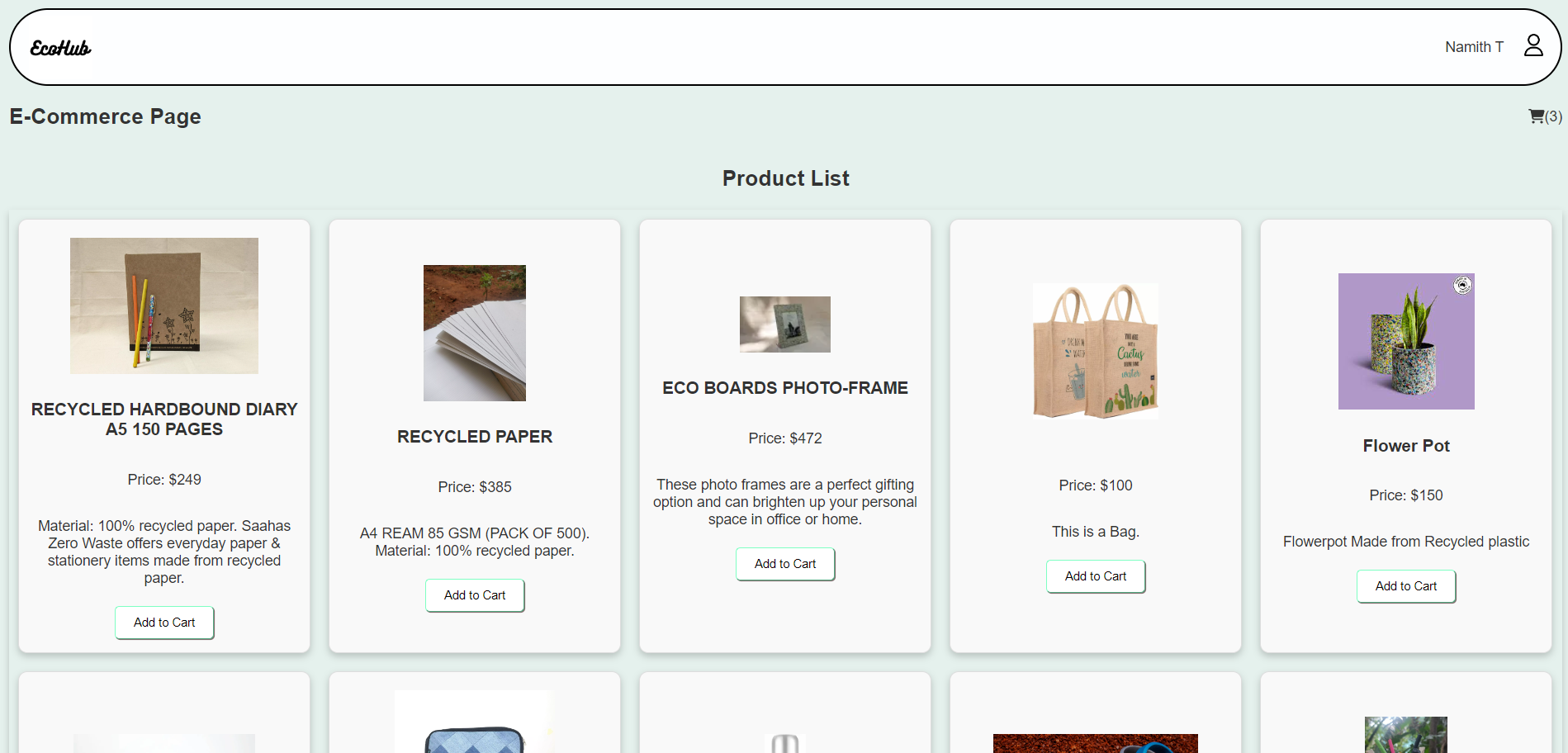
**9 Screen Shot**

**A screenshot of a web page

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

****

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**10 Conclusion**

The EcoHub project represents a significant endeavor aimed at promoting environmental sustainability and social responsibility through the utilization of technology. Throughout the development process, our team focused on creating a platform that facilitates waste management, promotes eco-friendly practices, and fosters community engagement.

By providing users with a convenient and accessible platform to request waste collection services, browse eco-friendly products, and contribute to environmental initiatives, EcoHub aims to empower individuals and communities to make a positive impact on the planet.

Throughout the development lifecycle, we encountered various challenges, made critical design decisions, and collaborated closely to deliver a robust and user-friendly application. By leveraging modern technologies such as React.js, Firebase, and cloud services, we were able to create a scalable and efficient solution that meets the needs of both users and administrators.

Continuous improvement, iteration, and adaptation will be essential to ensure the platform remains relevant, effective, and impactful in the long term. We are committed to listening to user feedback, addressing emerging challenges, and evolving the platform to meet the evolving needs of our users.

**11. Bibliography**

1. React Documentation. Retrieved from: https://reactjs.org/docs/getting-started.html
2. Firebase Documentation. Retrieved from: <https://firebase.google.com/docs>
3. MDN Web Docs - JavaScript. Retrieved from: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
4. CSS Tricks - CSS Documentation. Retrieved from: <https://css-tricks.com/>
5. W3Schools - HTML Tutorial. Retrieved from: https://www.w3schools.com/html/
6. "Environmental Sustainability." United Nations Environment Programme. Retrieved from: https://www.unep.org/explore-topics/sustainable-development-goals/why-do-sustainable-development-goals-matter/environmental
7. "Waste Management: A Guide to Environmental Conservation." Environmental Protection Agency. Retrieved from: <https://www.epa.gov/report-environment/waste-management-overview-and-resources>
8. "Eco-Friendly Practices for a Sustainable Future." World Wide Fund for Nature (WWF). Retrieved from: https://www.worldwildlife.org/stories/eco-friendly-practices-for-a-sustainable-future
9. "The Impact of Technology on Environmental Sustainability." Forbes. Retrieved from: https://www.forbes.com/sites/forbestechcouncil/2021/04/12/the-impact-of-technology-on-environmental-sustainability/?sh=360b188166b8
10. "E-commerce and Its Environmental Impact." GreenMatch. Retrieved from: https://www.greenmatch.co.uk/blog/2016/08/how-ecommerce-affects-the-environment
11. "Social Responsibility: A Key Component of Sustainable Business." Harvard Business Review. Retrieved from: <https://hbr.org/2020/01/social-responsibility-a-key-component-of-sustainable-business>